Bases de données avancées
Projet

Modalités

Le projet est à réaliser en binômes.

Une modélisation préliminaire (avec document explicatif) doit être rendue sur Moodle le 15 mars au plus tard. Avant de procéder à la réalisation/implémentation du projet, la modélisation doit être validée pendant un courte pré-soutenance qui sera organisée la semaine du 17 mars.

Le projet entier sera à rendre avant la session 1 d'examen (entre le 9 et 22 mai) ; les modalités définitives vous seront communiquées ultérieurement.

Présentation générale

Ce projet vise à concevoir un système de réservation de billets pour des événements comportant plusieurs sous-événements, et/ou avec une gestion avancée des règles de réservation pour les utilisateurs. Le modèle de données devra refléter les différentes phases du processus de réservation, incluant par exemple :

- une phase de pré-réservation,
- l'attribution de créneaux de connexion pour effectuer les achats
- la limitation du nombre de places par utilisateur en fonction de leur statut (extérieur, abonné, retraité, invité, etc ...)
- la politique mise en place par les établissements (par exemple des quotas)
- l'échange de billets,
- une totale traçabilité des transactions
- ...

Vous produirez un modèle de données détaillé, un ensemble de requêtes SQL significatif, ainsi que des procédures en PL/pgSQL permettant la gestion avancée des transactions, avec des jeux de tests sur des situations que vous imaginerez. Vous vous interrogerez également sur les stratégies d'optimisation pour améliorer les performances du système.

Réservation de billets

Pour vous permettre de distinguer vos travaux les uns des autres, nous souhaitons que vous vous inspiriez de la façon dont sont traités les réservations dans des situations concrètes que vous avez pu rencontrer. Dans cette présentation nous illustrons notre propos en prenant pour exemple parfois les jeux olympiques et parfois les abonnements à l'opéra, mais il est préférable que vous en trouviez d'autres : drops, préfecture, réservation de salle, ... Il faut respecter les deux contraintes : le système doit comporter des règles de réservation complexes, et vous devez vous préoccuper des problèmes liés à la gestion du trafic des connexions.
Votre schéma relationnel inclura assez naturellement les entités suivantes. Les attributs sont donnés à titre d'exemple et peuvent varier selon le contexte.

- Utilisateur avec id, nom, email, date d'inscription, statut VIP ou non, historique de réservation, points de fidélité, ...
- Evénement avec id, nom, date, lieu, type, capacité maximale, catégorie d'accès, ...
- La notion de sous-événement (s'il y a lieu)
- La notion de formule d'abonnement (s'il y a lieu)
- Les billets
- Les règles de limitation associées à un type d'utilisateur. Elles peuvent être globale (sur le nb de place totale), mais pourraient être aussi des combinaisons booléennes : 3 places max pour les finales, pas de superposition d'horaire, au choix : entre basket ou hand mais pas les deux, etc ..

Par exemple à l'opéra les choses se passent ainsi : Les anciens abonnés ont une priorité pour un réabonnement c'est à dire une période avancée, par exemple le 15 mars. Les autres attendrons le 15 avril. Le jour où les abonnements sont mis en vente, une salle d'attente est mise en place pour gérer le flux de connexions. Il existe plusieurs formules d'abonnement, par exemple, l'abonnement libre 4 spectacles et plus, et l'abonnement libre 6 spectacles et plus, qui donne droit à 10% de réduction sur l'ensemble des places. Toutes les places dans un abonnement sont prises dans une catégorie de places (1-8). Le prix est déterminé par la catégorie de place et la catégorie de spectacle.
Ne tentez pas d'implémenter toutes les règles de gestion du thème concret que vous aurez choisi de traiter. Contentez-vous d'un sous-ensemble représentatif des différents cas de figure. Faites une modélisation la plus complète possible, sans dépasser les 10 à 12 entités, et précisez quel sous-ensemble vous implémenterez.
Vous devrez produire un Modèle Conceptuel des Données détaillé, accompagné d'un schéma relationnel normalisé. **Attention à la syntaxe graphique !** Suivez impérativement celle que vous avez vu en L3, qui nous sert de référence, et que nous avons rappelée au TD1.
Dans votre rapport de modélisation, mettez toutes les clefs candidates, toutes les dépendances fonctionnelles non-triviales, et toutes les contraintes. Schématisez les informations sous forme de tableaux. Pour les contraintes, indiquez s'il faut les mettre comme contrainte CHECK ou s'il faudra implémenter un TRIGGER. Évitez les clefs auto-incrémentées, utilisez plutôt les clefs naturelles lorsque c'est possible : ce projet est un exercice académique.

Gestion du flux

Pour gérer le flux des demandes de réservation, plusieurs stratégies peuvent être mises en place. Un exemple récent est le système utilisé pour la billetterie des Jeux Olympiques qui a proposé l'accès aux billets en plusieurs phases distinctes.
Nous le décrivons (grossièrement) ci-dessous afin que vous puissiez vous en inspirer. Ce que nous vous demandons dans ce projet c'est de développer **une seule** stratégie de ce type. Vous êtes libre de proposer des variations.
Phase 1 – Préinscription : Une première vague recense d'abord les clients potentiels en leur demandant de se signaler à l'avance. Puis les inscrits sont mélangés et sélectionnés aléatoirement pour définir des groupes qui pourront accéder à des créneaux d'achat successif d'une certaine durée (ex. 24 heures) sur une période fixée (ex. du 1er au 7 mai). La taille de ces groupes dépend naturellement du nombre d'inscrits, la période, elle, est fixée à priori à la création de l'évènement.

Phase 2 – Deuxième ouverture : Lors de la seconde phase, un nouvel ensemble de billets est remis en vente. Les acheteurs sont à nouveau sélectionnés et répartis en blocs en tenant compte d'un autre critère (disons leur ordre d'inscription initial). Dans cette phase le temps de réservation est significativement plus court (par exemple 2h en journée sur 5 jours).
Phase 3 – Vente finale : Lors de la dernière phase, tous les billets restants sont mis en vente libre. L'accès est géré par des files d'attente des connexions avec une priorité attribuée selon le statut de l'acheteur (ex. abonnés fidèles, clients VIP, etc.). Pour éviter le déni de service, seuls une centaines portion de la file d'attente pourra poursuivre ses transactions pendant une période courte (time-out courts après 20 minutes par exemple).
Précaution simple : Pour lutter contre les robots, un captcha peut être mis en oeuvre à la connexion.

Requêtes SQL et implémentation en PL/pgSQL

Nous souhaitons que vous illustriez l'intérêt de votre modélisation avec quelques situations. Sans vous limiter ni vous contraindre à celles que nous vous présentons, voici quelques suggestions. Pour en faciliter l'utilisation, vous pourrez les rédiger en écrivant des PREPARE ou des fonctions plpgsql

- Insérer de nouveaux utilisateurs et événements.
- Récupérer les événements et sous-événements disponibles pour un utilisateur donné en fonction de son profil et de ses préférences. Une requête paramétrée peut être utile ici.
- Effectuer une pré-réservation (panier) pour un sous-événement en vérifiant la disponibilité et les limitations applicables.
- Annuler une réservation.
- Confirmer une réservation en s'assurant de la cohérence des données.

On peut aussi préparer des requêtes plus complexes, dont voici quelques exemples.

- Générer des statistiques sur les comportements de réservation (recherche de robots ou d'acheteurs suspects qui pourraient faire du marché noir)
- Gérer les créneaux de connexion des utilisateurs en fonction de leur statut, de leur historique et de la charge du système.
- Combiner les éléments précédent sous forme de transactions pour s'assurer de leur atomicité et de leur bonne annulation.
- La gestion automatique des créneaux de réservation
- Un système d'échange ou de revente de place.
- Un mécanisme de validation automatique des paiements et d'attribution des billets numériques.

Mise en oeuvre

Le développement de ce projet comporte deux grandes étapes.

Présoutenance

La première consiste en une modélisation suffisamment riche pour qu'elle soit intéressante. Vous pouvez modéliser plus de choses que vous n'en réaliserez ensuite. Il s'agit d'un travail préalable que vous présenterez et sur lequel nous vous donnerons un premier avis. Cette modélisation pourra être faite sur papier à la main [1] , mais elle devra impérativement respecter le langage graphique utilisé en cours (et pas un autre). Nous vous l'avons déjà dit un peu plus haut, mais il est essentiel d'être bien compris sur ce point !
A la seconde étape, celle de la réalisation, vous pourrez vous restreindre à une partie raisonnable de votre diagramme. Pour vous donner un ordre de grandeur de ce qui est attendu, le nombre de tables résultantes ne devrait pas dépasser une douzaine, suivant les choix de modélisation et les simplifications entre le modèle et l'implémentation.
Voici les modalités de rendu :

- créez un projet sur `moule.informatique.univ-paris-diderot.fr` et invitez les deux chargés de TP : Sophie Laplante et Yan Jurski. Votre `README` contiendra les noms et prénoms des auteurs du projet.
- un répertoire `MODELISATION` contiendra votre `diagramme.pdf` qui pourra être une photo d'une document papier du moment qu'il est parfaitement lisible. Si vous en avez plusieurs, donnez leur des noms clairs.
  Dans un document annexe `modelisation.pdf` il vous faudra décrire le système de réservation choisi et identifier un maximum de contraintes d'intégrité et de règles de gestion et nous préciser comment chacune pourra être gérée (CHECK, UNIQUE, trigger, programmation). Notez y toutes les hypothèses retenues, notamment les dépendances fonctionnelles et les hypothèses concernant les cardinalités.

Seuls les documents au format PDF seront acceptés.

Réalisation

Dans la deuxième étape vous créerez et alimenterez les tables (qui doivent correspondre à votre modèle), puis vous préparez des démonstrations et les étudierez.
Privilégiez le peuplement des tables à l'aide de fichiers csv en utilisant `COPY`. Si vous trouvez des données publiques pertinentes indiquez en la source. Vous pouvez aussi utiliser des générateurs de données (d'identités, de connexions etc ...) en les écrivant dans le langage de votre choix.
Dans le répertoire `CREATION` :

- vous déposerez des fichiers comme : `create_table.sql`, `insert_data.sql`, les fichiers csv de données, et éventuellement vos scripts de génération
- un fichier `create_triggers.sql` isolera la création des triggers et de leurs fonctions

Dans le répertoire `ETUDES`

- vous déposerez les scénarios (sous forme de script SQL) qui vous seront utiles pour votre soutenance. Il devrait y avoir quelques requêtes, des `Prepare` ou des fonctions, et puis des illustrations plus globales qui simulent des connexions en parallèle. N'oubliez pas de les commenter. Les scripts doivent pouvoir être exécutés par les chargés de TP donc pensez à inclure un fichier README pour expliquer comment et pourquoi lancer les différents fichiers.

1. vous les peaufinerez plus tard, par exemple à l'aide de `https://app.diagrams.net/`

Un mot sur la gestion du temps

Il ne serait pas réaliste d'utiliser le temps système pour vos démonstrations. C'est pourquoi vous utiliserez la table suivante lorsque vous voudrez connaître l'heure et le jour.

```
CREATE TABLE TEMPS (
 jour INTEGER NOT NULL,
 heure INTEGER NOT NULL CHECK (heure >= 0 AND heure < 24)
);

CREATE OR REPLACE FUNCTION controle_temps() RETURNS TRIGGER AS $$
BEGIN
 IF TG_OPNAME = 'UPDATE' THEN
  IF NEW.jour < OLD.jour THEN RAISE EXCEPTION 'Impossible';
  ELSIF NEW.jour = OLD.jour AND NEW.heure < OLD.heure
    THEN RAISE EXCEPTION 'Impossible';
  END IF;
 ELSIF TG_OPNAME = 'INSERT' THEN
  IF (SELECT COUNT(*) FROM TEMPS) > 0 THEN RAISE EXCEPTION 'Impossible';
  END IF;
 END IF;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER controle_temps_trigger BEFORE UPDATE OR INSERT ON TEMPS
FOR EACH ROW
 EXECUTE PROCEDURE controle_temps();
```

Soutenance et rapport

Le rapport de projet devra rappeler la modélisation en soulignant les éventuels changements par rapport à la modélisation présentée. Vous devez détailler :

- pour chaque contrainte, comment et où elle a été traitée
- les scenarios que vous présenterez à la soutenance
- comment les transactions sont gérées
- les index mis en place, avec brève justification.

Pensez à inclure quelques requêtes complexes.

La soutenance se déroulera par binôme, mais la notation et les questions pourront être individualisées. Vous devrez chacun maîtriser l'ensemble de ce qui est présenté, quelle que soit la façon dont vous vous êtes réparti le travail.

Toute utilisation d'IA générative est à proscrire. Si vous y avez recours, cela doit être clairement indiqué, tout comme les sources extérieures (recherches web par exemple, si vous avez repris du code directement).

Votre prestation sera tout aussi importante que la nature du travail rendu : il s'agit d'apprécier ce que vous savez faire. Vous pourrez être amené à modifier une partie de ce que vous avez fait pour que nous puissions constater votre réactivité. Nous devrions pouvoir enchaîner rapidement les questions, retrouver des lignes de codes très rapidement etc... Cela n'est possible que si vous avez pensé à de nombreux cas de figure, conservé et ordonné vos propres tests.

Concernant les transactions, vous pourrez ouvrir 3 consoles. Une pour chaque utilisateur

qui se connecte en parallèle, et une troisième pour permettre de simuler le passage du temps.